Ligação Dinâmica

Marília Felippe Chiozo

Novembro de 2007

Roteiro

O quê?

- Ligação dinâmica: mecanismo através do qual um programa pode ser executado mesmo que seu código resida em múltiplos arquivos.
- Um executável e um número arbitrário de *bibliotecas*.
- Através de um *loader*, o sistema operacional localiza as bibliotecas e resolve as referências pendentes.
- Dependente de arquitetura e formato de objeto.

O quê?

- Ligação dinâmica: mecanismo através do qual um programa pode ser executado mesmo que seu código resida em múltiplos arquivos.
- Um executável e um número arbitrário de *bibliotecas*.
- Através de um *loader*, o sistema operacional localiza as bibliotecas e resolve as referências pendentes.
- Dependente de arquitetura e formato de objeto.

O quê?

- Ligação dinâmica: mecanismo através do qual um programa pode ser executado mesmo que seu código resida em múltiplos arquivos.
- Um executável e um número arbitrário de *bibliotecas*.
- Através de um *loader*, o sistema operacional localiza as bibliotecas e resolve as referências pendentes.
- Dependente de arquitetura e formato de objeto.

O quê?

- Ligação dinâmica: mecanismo através do qual um programa pode ser executado mesmo que seu código resida em múltiplos arquivos.
- Um executável e um número arbitrário de *bibliotecas*.
- Através de um *loader*, o sistema operacional localiza as bibliotecas e resolve as referências pendentes.
- Dependente de arquitetura e formato de objeto.

Por que sim?

- Economia de espaço: evita a replicação desnecessária de código que é utilizado por vários programas.
- Facilidade de manutenção: tomadas as devidas precauções, apenas a biblioteca requer atualização.

Por que sim?

- Economia de espaço: evita a replicação desnecessária de código que é utilizado por vários programas.
- Facilidade de manutenção: tomadas as devidas precauções, apenas a biblioteca requer atualização.

Por que não?

- Nem sempre é vantajoso perder a autonomia de certos executáveis.
- A carga de um programa ligado dinamicamente é mais custosa do que a de um ligado estaticamente.
- A facilidade de manutenção proporcionada por bibliotecas compartilhadas pode se voltar contra programadores e usuários.

Por que não?

- Nem sempre é vantajoso perder a autonomia de certos executáveis.
- A carga de um programa ligado dinamicamente é mais custosa do que a de um ligado estaticamente.
- A facilidade de manutenção proporcionada por bibliotecas compartilhadas pode se voltar contra programadores e usuários.

Por que não?

- Nem sempre é vantajoso perder a autonomia de certos executáveis.
- A carga de um programa ligado dinamicamente é mais custosa do que a de um ligado estaticamente.
- A facilidade de manutenção proporcionada por bibliotecas compartilhadas pode se voltar contra programadores e usuários.

Relocação

- Ocorre na linkagem durante o processo de compilação, entre outras ocasiões.
- Substitui endereços nos arquivos de entrada por endereços utilizáveis no programa compilado.
- Dependente de arquitetura e formato de objeto.

Relocação

- Ocorre na linkagem durante o processo de compilação, entre outras ocasiões.
- Substitui endereços nos arquivos de entrada por endereços utilizáveis no programa compilado.
- Dependente de arquitetura e formato de objeto.

Relocação

- Ocorre na linkagem durante o processo de compilação, entre outras ocasiões.
- Substitui endereços nos arquivos de entrada por endereços utilizáveis no programa compilado.
- Dependente de arquitetura e formato de objeto.

Ligação dinâmica (ELF/*nix)

- O sistema operacional utiliza o suporte a interpretador do ELF para carregar o *loader*.
- O *loader* se inicializa e procede à carga das bibliotecas.
- O *loader* inicializa as bibliotecas carregadas.
- As bibliotecas não necessitam de relocação devido ao uso de código independente de posição.
- A resolução dos endereços de funções externas é adiada até a primeira chamada. (Ponteiros armazenados em espaço de dados.)

Ligação dinâmica (ELF/*nix)

- O sistema operacional utiliza o suporte a interpretador do ELF para carregar o *loader*.
- O *loader* se inicializa e procede à carga das bibliotecas.
- O *loader* inicializa as bibliotecas carregadas.
- As bibliotecas não necessitam de relocação devido ao uso de código independente de posição.
- A resolução dos endereços de funções externas é adiada até a primeira chamada. (Ponteiros armazenados em espaço de dados.)

Ligação dinâmica (ELF/*nix)

- O sistema operacional utiliza o suporte a interpretador do ELF para carregar o *loader*.
- O *loader* se inicializa e procede à carga das bibliotecas.
- O *loader* inicializa as bibliotecas carregadas.
- As bibliotecas não necessitam de relocação devido ao uso de código independente de posição.
- A resolução dos endereços de funções externas é adiada até a primeira chamada. (Ponteiros armazenados em espaço de dados.)

Ligação dinâmica (ELF/*nix)

- O sistema operacional utiliza o suporte a interpretador do ELF para carregar o *loader*.
- O *loader* se inicializa e procede à carga das bibliotecas.
- O *loader* inicializa as bibliotecas carregadas.
- As bibliotecas não necessitam de relocação devido ao uso de código independente de posição.
- A resolução dos endereços de funções externas é adiada até a primeira chamada. (Ponteiros armazenados em espaço de dados.)

Ligação dinâmica (ELF/*nix)

- O sistema operacional utiliza o suporte a interpretador do ELF para carregar o *loader*.
- O *loader* se inicializa e procede à carga das bibliotecas.
- O *loader* inicializa as bibliotecas carregadas.
- As bibliotecas não necessitam de relocação devido ao uso de código independente de posição.
- A resolução dos endereços de funções externas é adiada até a primeira chamada. (Ponteiros armazenados em espaço de dados.)

Ligação dinâmica (PE/Windows)

Diferenças do esquema anterior:

- O *loader* é fisicamente parte do kernel.
- Uso de *prelinking* para tentar evitar relocação de DLLs.
- Não há código independente de posição.

Ligação dinâmica (PE/Windows)

Diferenças do esquema anterior:

- O *loader* é fisicamente parte do kernel.
- Uso de *prelinking* para tentar evitar relocação de DLLs.
- Não há código independente de posição.

Ligação dinâmica (PE/Windows)

Diferenças do esquema anterior:

- O *loader* é fisicamente parte do kernel.
- Uso de *prelinking* para tentar evitar relocação de DLLs.
- Não há código independente de posição.

Referências

[1] Levine, J. R. *Linkers and Loaders* (manuscrito). 1999. URL: http://www.iecc.com/linker/
[2] Vários autores *ld.so man page*. In *Linux Programmer's Manual*. 16 de dezembro de 2001.
[3] Vários autores *Library (computing)*. Artigo da Wikipedia. 2007. URL: http://en.wikipedia.org/wiki/Library_(computer_science)